Dealing with Heterogeneity in an Active-based Multicast Congestion Avoidance Protocol

Moufida Maimour, Cong-Duc Pham

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Dealing with Heterogeneity in an Active-based Multicast Congestion Avoidance Protocol

Moufida Maimour, UCBL/INRIA/LIP

Cong-Duc Pham, UCBL/INRIA/LIP

N° 4796

28th March 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4796--FR+ENG

RHÔNE-ALPES

Dealing with Heterogeneity in an Active-based Multicast Congestion Avoidance Protocol

Moufida Maimour, UCBL/INRIA/LIP
Cong-Duc Pham, UCBL/INRIA/LIP

Thème 1 — Réseaux et systèmes
Projet RESO

Rapport de recherche n° 4796 — 28th March 2003 — 18 pages

Abstract: Many of the proposed multicast congestion avoidance algorithms are single-rate where heterogeneity is accommodated by adjusting the transmission rate as a response to the worst receiver in the group. Due to the Internet heterogeneity, a single-rate congestion control affects the overall satisfaction of a multicast session receivers. In this report, we propose a multi-rate replicated scheme where some receivers (instead of the source) are designated to perform data replication for other receivers with lower capacity. To be more scalable and to minimize the bandwidth consumption due to data replication, the partitioning algorithm is performed on-the-fly by the routers depending on the feedbacks they receive. Neither a prior estimation of the receivers capacity is necessary nor a complex computation is required to execute our partitioning algorithm.

Key-words: Reliable multicast, Congestion control, Heterogeneity, Active networks

This work is supported in part by the french ACI Grid program and by a grant from ANVAR-EZUS Lyon.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Prise en charge de l'hétérogénéité par un protocole de contrôle de congestion en multicast

Résumé : Traditionnelement, les algorithmes d'évitement de la congestion en multicast prennent en charge les récepteurs hétérogènes en adaptant le débit de la source à la capacité du récepteur le plus congestionné. De ce fait, les autres récepteurs ne sont pas *satisfaits* à cause de cette pénalisation. Dans ce rapport, nous proposons un schéma multi-débits où certains récepteurs (au lieu de la source) sont élus pour faire la réplication des paquets de données à d'autres récepteurs qui ont une capacité inférieure. Pour assurer le passage à l'échelle et diminuer la consommation de la bande passante dûe à la réplication des données, un algorithm de partitionnement des récepteurs est exécuté à-la-volée par les routeurs en fonction des messages de contrôle reçus. Ni une estimation à priori de la capacité des récepteurs est nécessaire, ni un calcul complexe est requis pour exécuter notre algorithm de partitionnement.

Mots-clés : Multicast fiable, Contrôle de congestion, Héterogénéité, Réseaux actifs

1 Introduction

The problem of reliability in multicast has been extensively studied and many proposals exist in the literature [3, 15, 17, 1, 14, 8, 9]. However congestion appears to be the most common reason for packet loss in the Internet. Any reliable multicast protocol requires congestion control to be addressed. We proposed a framework for bulk data distribution with two components, a reliable protocol, DyRAM (Dynamic Replier Active reliable Multicast) and a congestion avoidance protocol, AMCA (Active-based Multicast Congestion Avoidance algorithm) [11, 12]. AMCA was targeted to be used in conjunction with DyRAM but can be easily used with any other active reliable multicast protocol in addition to unicast as an enhancement of TCP Vegas.

AMCA is a single-rate congestion avoidance protocol where heterogeneity is accomodated by adjusting its rate in response to the most congested pathin the multicast tree. This latter is dynamically determined based on round trip time (RTT) variation estimated in a per-hop manner. Transmitting with a rate which matches the slowest receiver will limit the throughput of other receivers and thus their *satisfaction*. In a multicast session, a multi-rate mechanism can improve the receivers' satisfaction. In fact, receivers with different needs can be served at a rate closer to their needs rather than having to match the speed of the slowest receiver. In a multi-rate session, the multicast source can transmit at different rates either through a hierarchical scheme (layering) [4, 10, 16] or a replicated scheme (destination set grouping, DSG [6]). In layered multicast, each receiver controls the rate at which it receives data, usually by using multiple multicast groups. The receivers join and leave groups depending on their path congestion state so the amount of data being received is always appropriate. In a replicated scheme, the source splits the receivers into subgroups of similar capacities. Afterwards, it sends a separate flow to every subgroup with the appropriate rate. Layering schemes provide more economical bandwidth usage than DSG schemes, however layering is more complicated and requires efficient hierarchical encoding/decoding algorithms and synchronization among different layers.

In order to avoid the complexity inherent to a layered scheme especially for the case of a fully reliable multicast, we propose to use a replicated scheme. However our scheme differs from the existing replicated schemes in the following main points (*i*) the partitioning of the receivers is performed by the routers instead of the source, (*ii*) the partitioning algorithm is executed on-the-fly depending on the RTT variations instead of the receivers' isolated rates[1] which are very difficult to be estimated in the current Internet, and (*iii*) the data replication is no longer the responsibility of the source. More precisely, our solution consists in a distributed approach where some receivers (*replicators*) contribute in the replication of the data flow with an appropriate rate to other receivers of lower capacity. In this way, a *regulation tree* is built with the source as the root, the replicators as intermediate nodes and the remaining receivers as final nodes. In order to minimize bandwidth consumption due to data replication, the regulation tree is built in respect to the physical multicast tree. This is achieved by involving the routers in the regulation tree construction procedure. Every router performs a partition of its downstream links into subgroups and chooses a replicator for every subgroup formed. In addition to the construction of a regulation tree with a topology close to the physical multicast one, executing the partitioning algorithm at the routers instead of the source is more scalable since (*i*) every router performs a partitioning algorithm locally, (*ii*) there is no data replication at the source which still send only one data flow, and (*iii*) the transmission rate of the source is no longer dictated by the worst receiver in the whole multicast group.

The rest of this report is organized as follows. An overview of the DyRAM/AMCA framework is presented with more focus on the congestion avoidance mechanism in section 2. Section 3 presents a general algorithm for the construction of a regulation tree. Section 4 applies this algorithm for the extension of AMCA to support more heterogeneous receivers. Some simulation results are the object of section 5, and section 6 concludes.

2 DyRAM/AMCA Framework

DyRAM/AMCA is a framework for reliable multicast that deals with both reliability (DyRAM) and congestion avoidance (AMCA) components. This section gives an overview of the DyRAM/AMCA framework

[1] The isolated rate [5] is the rate that a receiver would obtain if unconstrained by the other receivers in the group, assuming max-min link sharing.

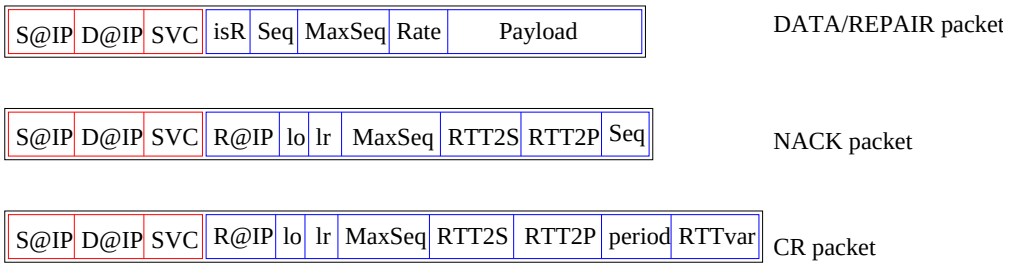


Figure 1: Packets structure.

before providing its extension to support more heterogeneous receivers. For more details about DyRAM and AMCA, the reader can refer respectively to [11] and [12].

DyRAM uses a recovery strategy based on a tree structure constructed on a per-packet basis with the assistance of routers. It uses a receiver-based local recovery where receivers are responsible for both the loss detection and the retransmission of repair packets when it is possible. The source sends data packets to the multicast address subscribed to by all the receivers. A receiver detects a loss by sequence gaps or timeouts. Upon the detection of a loss, a receiver sends immediately a NACK toward the source and sets a timer. A receiver will re-send a similar NACK when the requested repair has not been received within the timeout interval. On the reception of a NACK packet, the source sends the repair packet to the multicast address. In order to limit the processing overheads of duplicate NACKs and to avoid the corresponding retransmissions, the source, the active routers and the receivers (remember that DyRAM uses repliers among receivers) ignore similar NACKs for a certain period of time. In order to perform flow and congestion control as well as memory management in addition to a more efficient replier election, ACKs are piggybacked on the NACKs. In the absence of NACKs, ACKs are also periodically piggybacked on special messages called "Congestion Reports" (CRs). The DyRAM active services can be summarized as follows:

- the early loss detection of packet losses and the emission of the corresponding NACKs. This feature provides a low recovery delay.
- the NACK suppression in order to limit the NACK implosion problem.
- the subcast of the repair packets only to the relevant set of receivers that have experienced a loss. This helps to limit the scope of the repairs to the affected subtree.
- the dynamic replier election which consists in choosing a link as a replier to perform a local recovery from a receiver downstream instead of caching data packets at the routers. Dynamic election provides robustness to host and link failures.

For the purpose of congestion control, AMCA uses the active networking technology to perform on a per-section dialogue to probe for available bandwidth along a multicast tree. In this context a section is defined as the set of point-to-point links and traditional routers that connect two active routers or an active router to a terminal node (a receiver or the source). The solution uses the RTT variations experienced by every branch to estimate the congestion situation in the multicast tree. Every receiver sends a CR to the source every N packets received so the source can learn about the congestion situation in the multicast tree. A CR contains mainly information about RTT variation used by the source to adjust the rate of the transmission. The physical multicast tree is used to appropriately aggregate the RTT variations at intermediate nodes before they reach the source.

A DyRAM/AMCA packet contains in its header the multicast address, the source address and the active service identifier (SVC) to be performed on this packet (see figure 1). A data packet is labelled uniquely by a sequence number and contains a dedicated field (isR) to distinguish an original transmission from a repair. A data packet contains also the current emission rate which is mainly useful to be known by the receivers in order to correctly update timeouts. A NACK or a CR packet include in addition to their originator address, their last ordered (lo) and last received (lr) data packet. Moreover, a receiver will report its RTT to the

source and to its active router using the *RTT2S* and *RTT2P* fields contained in NACK and CR packets. Whereas a NACK contains the sequence number of the requested data packet, a CR contains two other fields which are the CR current period and an aggregated value of the RTT variation experienced by the subtree from which the CR is received.

For the purpose of flow control, every receiver includes in the $Maxseq$ field of a CR/NACK (figure 1), the maximum data packet sequence number that corresponds to its available buffering means. The $MaxSeq$ field in a CR or NACK packet is used by the source to control the transmission flow. An active router includes the minimum of the received $MaxSeq$ in the CR to be sent upstream to the source. The source then sends data packets with the current rate until the minimum $MaxSeq$ of all the receivers. The lo field is also used to release buffers at both the sender and the receivers. With the proposed mechanism of aggregating the CRs, the source ends-up by receiving the minimum lo of the whole group. This is an acknowledgement that all data packets up to lo have been correctly received by all the receivers. The source could then release buffers of data packets with sequence numbers up to lo. The source includes the current lo of the whole group in the $MaxSeq$ field of the data packets it sends. A receiver on the reception of a data packet would update its flow control window according to the $MaxSeq$ field of the received data packets. Moreover and since a receiver would keep data packets in order to be able to act as an eventual replier, this mechanism is helpful to release memory. A receiver could remove from its memory all the data packets with a sequence number less or equal to the $MaxSeq$ specified by the received data packet.

2.1 Router's Soft State

Within active routers, the different active services can be implemented simply by maintaining information about the data packets, NACKs and CRs received. This set of information is uniquely identified by the session source and the multicast address. For each received NACK, the router creates or simply updates an existing NACK state (NS) structure which is removed on the reception of the corresponding repair. This structure maintains for every nacked packet, a subcast list ($subList$) that contains the numbers of links from which a NACK has been received. DyRAM also enables the routers to keep track of the received data packets in order to quickly detect packet losses occurring from upstream and affecting all its downstream links. This can be done simply by maintaining a track list (TL) structure for each multicast session handled by the router. A TL has three components:

- $lastOrdered$ is the sequence number of the last data packet received in the order.
- $lastReceived$ is the sequence number of the last received data packet.
- $lostList$ contains the list of data packets not received by the router with sequence number greater than $lastOrdered$ and less than $lastReceived$. This list is empty when ($lastReceived < lastOrdered + 1$) and contains at least one element otherwise.

In order to perform quicker, accurate and optimized replier elections, we keep within a router a link state structure (LS) which contains for every downstream link three fields[2] which are :

- lo is the sequence number of the last data packet received in order on this link which corresponds to the last one received in order by all the receivers located downstream this link.
- lr is the sequence number of the last received data packet by this link which corresponds to the last received one by all the receivers located downstream this link.

2.2 RTT and RTT Variation Estimation

In our DyRAM/AMCA framework, many timeouts are set depending on RTT measures. Moreover, the associated congestion avoidance algorithm requires the estimation of RTT variations. In this section, we provide our mechanism used to estimate the RTTs and their variation in a scalable way. A straightforward solution to estimate the RTTs between each receiver and the source consists in sending ping messages

[2]These information are retrieved from the CRs and NACKs received during the multicast session.

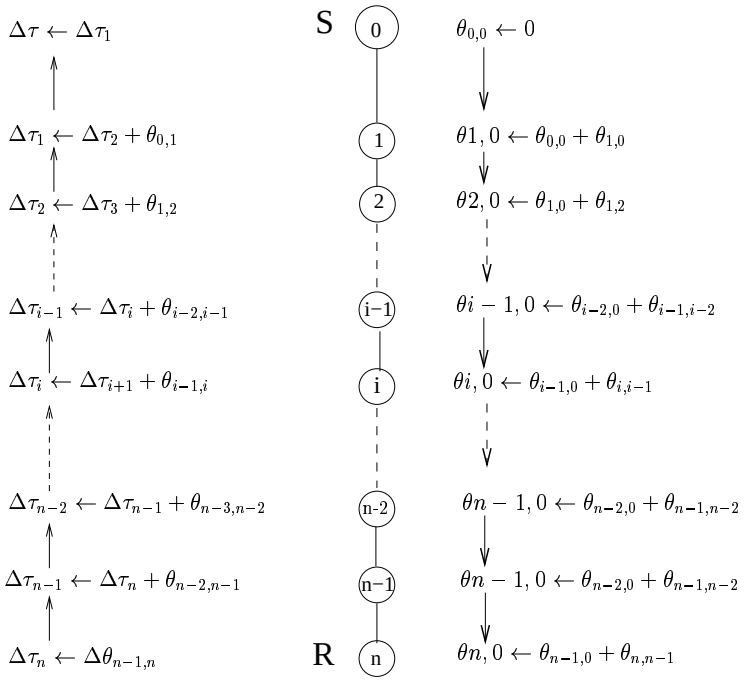


Figure 2: RTT estimation and CRs aggregation

periodically from the receivers to the source. This naive solution would however overload the source in the presence of a large number of receivers. Instead of estimating directly the RTTs between the source and each receiver, we begin by estimating the RTTs between every node and its parent in the multicast tree. To illustrate our method we show in figure 2, one linear path of the multicast tree that connects the sender (labeled by 0) to one of the receivers with label n. Intermediate nodes labeled from 1 to $n-1$ are the routers. The generalization of our method to the other receivers and intermediate nodes is straightforward.

Special messages called heartbeat messages (HB and HB_RESP) are periodically exchanged between each node i and its parent $(i-1)$ in the same way the ping/pong messages do. The first HB message is sent by the receiver. The reception of a HB by a router triggers the emission of a HB message to its parent in addition to responding with a HB_RESP message. Let $\theta_{i,j}$ be the computed RTT between the ith and the jth node). Consequently, $\theta_{i-1,i}$ is the RTT between the node i and its parent and $\theta_{0,i}$ is the RTT between a node i and the source.

A node i, in order to compute its RTT to the source, first computes its RTT to its parent node by sending a HB message timestamped with the emission time. The parent node, on the reception of the HB message, responds with a HB_RESP that contains its RTT to the source that has already been computed. Node i, on receipt of the HB_RESP message is then able to compute its RTT to the source as the sum of the RTT of its parent to the source and its own RTT to its parent:

$$\forall i > 0 : \theta_{0,i} = \theta_{0,i-1} + \theta_{i-1,i} \tag{1}$$

Initially (right side of figure 2), the source responds with a special message HB_RESP_S which contains its own RTT to itself (of course we have $\theta_{0,0} = 0$) in the data packets it sends. Node number 1 will update its $\theta_{0,1}$ using (1) and forwards it downstream with the newly computed RTT. Node number 2 will in turn update its $\theta_{0,2}$ using the same equation and forwards it downstream and so on (figure 2). Afterwards the use of the HB and HB_RESP messages will be sufficient to estimate the different RTTs.

In what follows, we note by $\Delta\theta_{i,j}$ the experienced RTT variation between the ith and the jth node. For the ith link that connects the $(i-1)$th and the ith node, the RTT variation is $\Delta\theta_{i-1,i}$. Additionally we note by $\Delta\tau_i$ the RTT variation of the $(i-1)$th intermediate node and the receiver. The overall experienced RTT

variation between the source and the receiver is $\Delta\tau = \Delta\tau_1$. Initially, a receiver (left side of figure 2) computes the RTT variation $\Delta\theta_{n-1,n}$ of its upstream link (number n) using the last two RTT measures to its parent with the mechanism described earlier in this section. Afterwards the receiver includes the computed RTT variation in the CR to be sent to the source. The receiver's parent (i.e. node $n-1$) adds the RTT variation reported by the receiver and its own RTT variation to its upstream node to obtain the RTT variation of its parent ($(n-2)$th node) to the receiver:

$$\Delta\tau_{n-1} = \Delta\tau_n + \Delta\theta_{n-2,n-1}$$

Node $(n-1)$ will then report $\Delta\tau_{n-1}$ in the *RTTvar* field of the CR. In this way, a parent node i, on receipt of a CR from node $(i+1)$ adds its RTT variation to node $(i-1)$ to the received one before forwarding the CR with the newly computed RTT variation:

$$\Delta\tau_i = \Delta\tau_{i+1} + \Delta\theta_{i-1,i}$$

The sender will end up by receiving the RTT variation experienced on the whole linear connection $\Delta\tau$.

2.3 Congestion Feedback Suppression/Aggregation

To achieve scalability, a mechanism of aggregation/suppression is performed. We use the physical multicast tree to hierarchically aggregating feedbacks at the intermediate nodes (routers). A suppression mechanism is performed on NACKs thus allowing just one NACK per loss to be sent upstream. Since we do not suppose that all the routers are active, more than one NACK for the same loss could be received by the source. To avoid reacting to duplicate NACKs, the source reacts to the first one and ignores the subsequent ones for a given period of time which depends on the multicast tree structure.

The CRs are aggregated at the intermediate nodes. An active router aggregates the received CRs from the downstream links in one CR to be forwarded upstream. The aggregated CR contains the sequence number of the minimum last ordered and the maximum last received data packets among those reported by the CRs received from downstream. For the aggregation of the RTT variations reported by the CRs from downstream in the case of a multi-path connection, we illustrate our solution with the simple two-level tree depicted in figure 3. We consider two receivers R_1 and R_2 connected to the source S via one active router A. Receivers R_1 and R_2 send CRs to the active router with their respective RTT variations $\Delta\tau_1$ and $\Delta\tau_2$. Once these children CRs are received, the active router A sends its CR to the source with $\Delta\tau$ computed using:

$$\Delta\tau = \Delta\tau_{up} + Max(\Delta\tau_1, \Delta\tau_2)$$

where $\Delta\tau_{up}$ is the RTT variation of the source link (A, S). This latter is extracted from the HB_RESP message sent by the source. Using this method in aggregating the RTT variation, the source ends up by receiving the RTT variation experienced by the worst end-to-end path of the multicast tree.

2.4 Rate Adjustment

In AMCA, a minimum and a maximum rates r_{min} and r_{max} are set by the application. Any receiver that could not support the minimum transmission rate has to leave the multicast session. Initially, the source starts to send data packets with a rate equal to the minimum rate r_{min}. Then it tries to increase its rate if no congestion indication is received without exceeding the maximum rate r_{max}. The source uses the information fed back in the CRs and NACKs to update its rate. The RTT variation field of every CR is used by the source to compute the queue size variation per packet Δq_p during the previous period. We have:

$$\Delta q_p = \frac{\Delta\tau}{\Delta\tau + T} \tag{2}$$

where Δ_τ is the RTT variation experienced by the worst end-to-end path of the multicast tree. The rate regulation at the source is also based on an other metric that estimates the number of data packets which are not acked yet by all the receivers Δq_a:

$$\Delta q_a = \max(0, N - (lo_{i+1} - lo_i)) \tag{3}$$

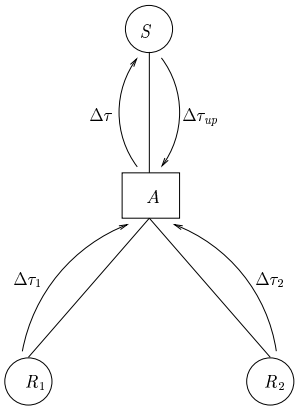


Figure 3: CRs aggregation in a multicast tree

where N is the number of packets received during the period value T. lo_{i+1} and lo_i are respectively the values of the last ordered field of the newly and previously received CR. Note that Δq_a gives a measure of the difference between the emission and the reception rate.

The source, on the reception of a CR, extracts the current period (T) and the RTT variation ($\Delta\tau$). Afterwards, it computes the queue size variation per packet Δq_p and the number of data packets not acked yet Δq_a using (2) and (3). The aim is to maintain $\Delta q_p < \epsilon$, where ϵ is a positive number to be chosen in $[0, 1]$ and Δq_a in $[\alpha, \beta]$, where α and β are similar to the two parameters of TCP-Vegas. During a phase similar to the TCP slow start, the source continues increasing its rate by S/RTT_{max} (bits per second) every time it receives a CR that indicates $\Delta q_p < \epsilon$ and $\Delta q_a < \beta$. Increasing the rate by S/RTT_{max} where RTT_{max} is the maximum experienced RTT among the receivers, is equivalent to adding 1 to the congestion window for the largest end-to-end connection of the multicast tree. This behavior makes our congestion avoidance algorithm fair with TCP from the beginning of the multicast session. The source enters the congestion avoidance phase when it receives a NACK or a CR with $\Delta q_p > \epsilon$ or $\Delta q_a > \beta$. On the receipt of a NACK, the source reduces the rate by half. Subsequent NACKs are ignored during a period estimated by the difference between the current RTT to the source of the farthest and the closest receiver in the multicast tree ($RTT_{max} - RTT_{min}$). To avoid decreasing dramatically the rate because of isolated losses, we check the lo and lr fields of the NACKs and reduce the rate only if $lr - lo \geq 3$, otherwise we just retransmit the corresponding repair. During the congestion avoidance phase, on the reception of a CR with a RTT variation $\Delta\tau$ and a period T, the source updates its rate depending on the current values of both Δq_p and Δq_a. When these two measures do not indicate congestion ($\Delta q_p \in [0, \epsilon]$ and $\Delta q_a < \beta$), the rate is multiplied by γ chosen slightly greater than 1 (say 1.025). In the other cases, the source updates its rate depending on the current values of both Δq_p and Δq_a. When these two measures do not indicate congestion ($\Delta q_p \in [0, \epsilon]$ and $\Delta q_a < \beta$), the rate is multiplied by γ_{inc} chosen slightly greater than 1. In the other cases, the rate is multiplied by $\gamma = T/(T + \Delta\tau)$[3]. When the emission rate is greater than the bottleneck rate, a queue will build up in the bottleneck link. This will be translated by a positive RTT variation $\Delta\tau$ and thus $\gamma < 1$. Multiplying by γ will decrease the transmission rate and this is the appropriate action from the source. In the reverse situation where the emission rate is less than the available bandwidth, $\Delta\tau$ is negative and multiplying by $\gamma > 1$ will increase the transmission rate. In addition to the proposed mechanisms, a severe congestion is detected if no feedback is received during a relatively long period which is set to twice the current period ($2T$). In this case the rate is dropped to its minimum value r_{min}.

[3]The rational behind this is shown in [12].

3 Accommodating Heterogeneity through a Regulation Tree

3.1 Background

Our approach is mainly based on a partitioning of the receivers into subgroups with similar capacities so their overall satisfaction is maximized. Our adopted partitioning algorithm is based on the *relative RTT variation* (noted $\Delta\dot{\tau}$), defined as the ratio of a measured RTT variation to the periodicity of this measure. We have $\Delta\dot{\tau} = \Delta\tau/T$ where $\Delta\tau$ is the RTT variation and T is the period duration. The properties and details of our partitioning algorithm are beyond the scope of this report, a deeper study of this algorithm is provided in [13]. In order to quantify the satisfaction of a receiver R_i in a multicast session, we use a utility function defined as follows [5, 7]:

$$U_i(r) = \frac{\min(r_i, r)}{\max(r_i, r)} \tag{4}$$

where r_i and r are respectively the isolated and the reception rate of receiver R_i. The receiver satisfaction is maximized when its reception rate is equal to its isolated rate, $U_i(r_i) = 1$. A receiver that receives data with a rate which is greater than its isolated rate could experience losses. In the opposite case, the receiver will also be unsatisfied since it has more available bandwidth. In a similar way to [5], we define the utility function of a single-rate multicast session as the weighted sum of the individual utility values of receivers in the session:

$$U(r) = \sum_{i=1}^{n} \alpha_i U_i(r) \tag{5}$$

subject to $\sum \alpha_i = 1$ and $\alpha_i \in [0,1], i = 1, \ldots, n$ where n is the number of the receivers in the multicast session. A multi-rate multicast session consists of one or more subgroups split from an original multicast session. The session utility function in this case is defined as the summation of the utility values obtained by all multicast subgroups, using the single-rate utility measure in each subgroup. More specifically, if a multicast session of receivers $\{R_1, R_2, \ldots, R_N\}$ is split into K subgroups $\{G_1, G_2, \ldots, G_K\}$ with different transmission rates $g_1, g_2, \ldots, g_K$, then

$$U(g_1, g_2, \ldots, g_K) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \alpha_{i,j} U_{i,j}(g_j) \tag{6}$$

subject to $\sum_{i,j} \alpha_{i,j} = 1$ and $\alpha_{i,j} \in [0,1]$. We have $\sum_j n_j = N$ where n_j is the number of the receivers in subgroup G_j. $U_{i,j}(g_j)$ and $\alpha_{i,j}$ are respectively the utility function and the weight associated to the ith receiver of the jth subgroup. Since we are concerned by a fully reliable multicast, the transmission rate g_j for the receivers in subgroup G_j has to match the minimum rate of the subgroup isolated rates, i.e. $\forall j, \quad g_j = \min_{i \in G_j} r_i$

3.2 Regulation Tree Construction

The regulation tree construction is a distributed process where each router performs locally a partitioning of its downstream links into subgroups and designates a replicator for every subgroup formed. Algorithm 1 shows how a router contributes in building such a regulation tree, on-the-fly depending on the RTT variation feedbacks it receives. We do not show the transmission rate adjustment in this algorithm since it is the role of the source and the replicators[4].

Initially, a router maintains a list P_0 that contains all of its downstream links. Every time, a downstream link experiences a relative RTT variation ($\Delta\dot{\tau}_j$) greater than a given parameter b, the router creates a new partition P_i with all the links that experienced a relative RTT variation greater than a parameter a ($a < b$) and selects a replicator Rep_i for this subgroup. We note by $Rd(P_i)$ the set of direct receivers located downstream of the P_i links. The function $Best(P_i)$ returns for subgroup P_i , the identity of a receiver from $Rd(P_i)$ which has the highest estimated capacity[5]. For instance, when the subgroup P_i is split from P_0, then

[4] A replicator as will be seen, has to adjust its replication rate depending on feedbacks it receives from its children receivers in the regulation tree.

[5] a receiver capacity is evaluated using a metric that will be presented in the context of AMCA, in section 4.1.

$Best(P_0)$ is elected as its replicator. Since this replicator may have been chosen for P_{i-1} in the previous split, $Best(P_i)$ is definitely elected as the P_{i-1}'s replicator (see algorithm 1).

Algorithm 1 Regulation tree construction at a router

Require: $N > 1$ and $a < b$
  $P_0 \leftarrow \{l_j, j = 1, \ldots, N\}$, the set of all the links downstream
  $i \leftarrow 1$
  Periodically,
  if $\exists j, l_j \in P_0$ such that $\Delta\dot{\tau}_j > b$ **then**
    $P_i \leftarrow \{l_j \in P_0, \Delta\dot{\tau}_j > a\}$
    $P_0 \leftarrow P_0 - P_i$
    $Rep_i \leftarrow Best(P_0)$
    if $i > 1$ **then**
      $Rep_{i-1} \leftarrow Best(P_i)$
    end if
    $i \leftarrow i + 1$
  end if
  until no split is possible

In our approach, a router forwards the source data packets only on the P_0 links. Every time, a new subgroup P_i is formed and the corresponding replicator Rep_i is selected, the router notifies this latter to start performing data replication[6]. A replicator Rep_i sends its replicated data packets to the receivers of its corresponding subgroup P_i. Feedbacks from subgroup P_i are sent to their corresponding replicator Rep_i while those arriving on the P_0 links are forwarded to the source. In this way the source transmission rate is dedicated by the worst receiver located downstream the P_0 links. Once the regulation tree is constructed, the source (and every replicator) sends data to its children with a rate that matches their minimum isolated rate without of course exceeding its reception rate (or equivalently, its parent transmission rate). That is, for an intermediate (replicator) node i:

$$rate_{trx}(i) = min\left(rate_{trx}\left(Parent(i)\right), min_j\{r_j, j \in Child(i)\}\right)$$

where $Parent(i)$ is the parent of node i, $Child(i)$ is the set of its children in the regulation tree and r_j is the isolated rate of receiver j.

3.3 Illustrative Examples

To illustrate our regulation tree construction algorithm, we consider a multicast session with seven subscribed receivers $\{R_0, R_1, R_2, R_3, R_4, R_5, R_6\}$ with respectively the following isolated rates $\{5, 6, 9, 11, 15, 19, 21\}$. All of these receivers are located downstream the same router A (see figure 4). If we take $a = 0.01$ and $b = 0.26$ which correspond to $\rho = 0.8$, executing algorithm 1 produces $\{\{R_0, R_1\}, \{R_2, R_3\}, \{R_4\}, \{R_5, R_6\}\}$ as a partition with the regulation tree shown in figure 4a. The resulting subgroups are $P_0 = \{R_5, R_6\}$, $P_1 = \{R_0, R_1\}$, $P_2 = \{R_2, R_3\}$, $P_3 = \{R_4\}$ with respectively the reception rates, $g_0 = 19$, $g_1 = 5$, $g_2 = 9$, $g_3 = 15$, which gives a session utility of 0.936 instead of 0.525 if no split is performed ($\forall i, \alpha_i = 1/7$). The selected replicator for the first split subgroup $P_1 = \{R_0, R_1\}$ is $Best(P_0) = R_6$. In the next split, when the second subgroup P_2 is formed with $Best(P_0) = R_6$ as the replicator then the P_1 replicator (according to the proposed algorithm) has to be changed to $Best(P_2) = R_3$ which is definitely elected as a replicator for P_1. When $P_3 = \{R_4\}$ is split, its chosen replicator is $Best(P_0) = R_6$ and $Best(P_3) = R_4$ is definitely selected as a replicator for P_2 ... etc. We can note that the built tree is not balanced. For instance, receiver 5 could be the replicator for P_2 as shown in figure 4(b). To have a more balanced tree, we can use an other rule for selecting the replicators. In fact, when partition P_i is formed, a replicator is chosen as follows:

[6] How this notification could be performed is addressed in section 4.1.

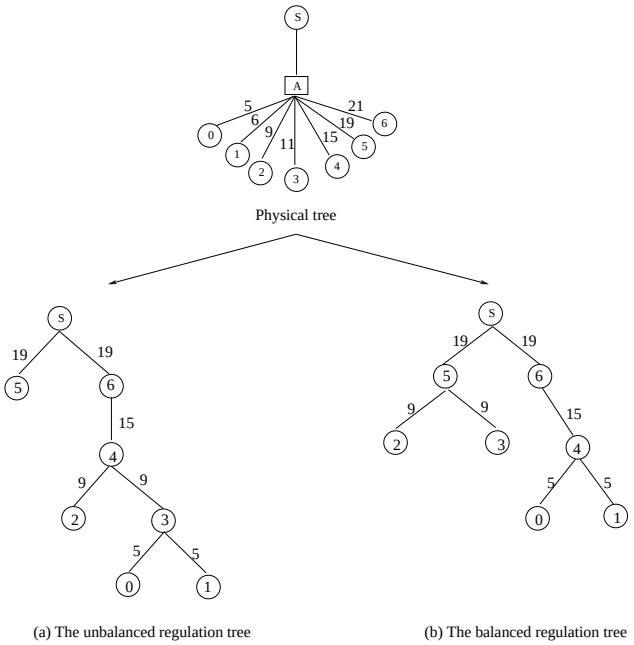


Figure 4: Simple construction tree.

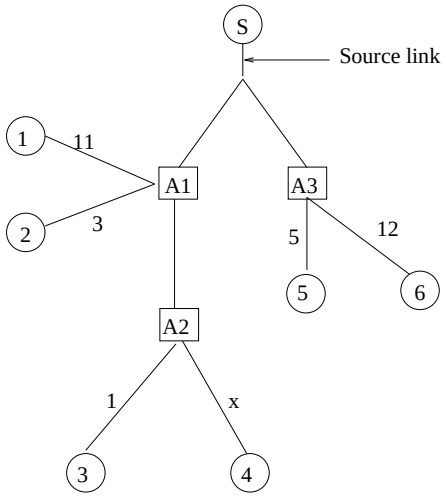


Figure 5: Example with a hierarchy of routers

$$Rep_i = Best\left(\bigcup_{k>i,k=0} P_k - \{Rep_k, k > i\} \right)$$

In order to understand our distributed partitioning mechanism, we consider one source multicasting to 6 receivers through 3 routers $A1$, $A2$ and $A3$ (figure 5). A maximum bandwidth is given for every link in the considered multicast tree. The link ($A2$,R_4) has a bandwidth of x which will be set either to 10 or 2 for the following. The main concern here is the behavior of router A_1 since it has two indirect receivers R_3 and R_4 in addition to two direct ones, R_1 and R_2. In what follows, a link will be designated by its downstream node in the multicast tree. For example the $A1$ downstream links that lead to router $A2$ and receiver R_1 are respectively noted $A2$ and R_1. We also use $P_{i,j}$ to designate the ith partition of router A_j. Suppose that

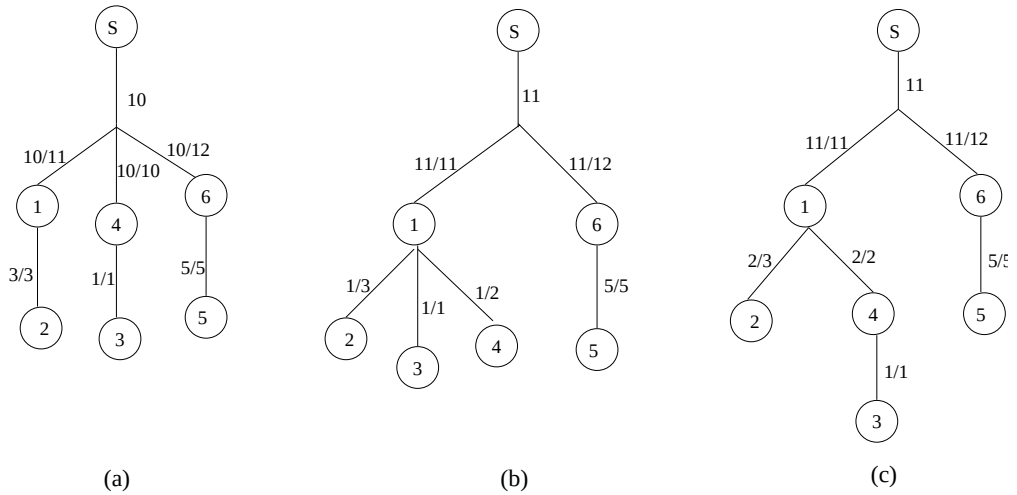


Figure 6: (a) $x = 10$, (b) $x = 2$ and $\rho < 0.5$, (c) $x = 2$ and $\rho > 0.5$

the maximum number of subgroups that could be built by a router is 2. If we set x to 10, then we get the following local partitions for the routers:

$$P_{0,1} = \{R_1, A2\}\ and\ P_{1,1} = \{R_2\} \qquad P_{0,2} = \{R_4\}\ and\ P_{1,2} = \{R_3\} \qquad P_{0,3} = \{R_6\}\ and\ P_{1,3} = \{R_5\}$$

which gives the following overall partition seen by the source (figure 6a):

$$P_0 = \{R_1, R_4, R_6\}, P_1 = \{R_2\}, P_2 = \{R_3\}, P_3 = \{R_5\} \tag{7}$$

This partition (7) gives a utility value, $U_a = (1/1 + 3/3 + 5/5 + 10/10 + 10/11 + 10/12)/6 = 0.957$ instead of 0.301 if no partitioning is performed ($\forall i, \alpha_i = 1/6$). In this latter case the source would transmit data at a rate equal to 1 instead of 10. If we consider that the partitioning is performed by the source instead of the routers, then the optimal partition with two subgroups is $P_0 = \{R_3, R_2, R_5\}$ and $P_1 = \{R_1, R_4, R_6\}$ which gives a utility value of $U'_a = 0.713 < U_a = 0.957$. In order to get a utility value equal to $U_a = 0.957$, a source-based partitioning requires four subgroups. In this case, the source link will be loaded with $10 + 1 + 3 + 5 = 19$ instead of 10 since the source will send four flows with rates 10, 1, 3 and 5. This gives $19/10 = 1.9$ of additional consumed bandwidth at the source link.

If x is set to 2, then depending on ρ, we get respectively for $\rho < 0.5$ and $\rho > 0.5$:

$$P_{0,1} = \{R_1\}\ and\ P_{1,1} = \{R_2, A2\} \qquad P_{0,2} = \{R_3, R_4\} \qquad P_{0,3} = \{R_6\}\ and\ P_{1,3} = \{R_5\}$$

and

$$P_{0,1} = \{R_1\}\ and\ P_{1,1} = \{R_2, A2\} \qquad P_{0,2} = \{R_4\}\ and\ P_{1,2} = \{R_3\} \qquad P_{0,3} = \{R_6\}\ and\ P_{1,3} = \{R_5\}$$

that gives respectively the following overall partitions:

$$P_0 = \{R_1, R_6\}, P_1 = \{R_2, R_3, R_4\}, P_2 = \{R_5\} \tag{8}$$

and

$$P_0 = \{R_1, R_6\}, P_1 = \{R_2, R_4\}, P_2 = \{R_3\}, P_3 = \{R_5\} \tag{9}$$

It is worth noting that when $x = 2$, the optimal source-based partition with two subgroups is $P_0 = \{R_2, R_4, R_3, R_5\}$ and $P_1 = \{R_1, R_6\}$ which gives a utility value of $U'_b = 0.667$ instead of $U_b = 0.792$ or $U_c = 0.931$ for the partitions (8) (figure 6b) and (9) (figure 6c).

4 AMCA Extension

In order to implement our regulation tree scheme, two approaches are possible. One approach could consist in the use of multiple multicast addresses, one per subgroup. Each replicator sends its data flow to a multicast address subscribed to by only its children in the regulation tree. In this case we need to implement a mechanism that would allow receivers (associated to a given replicator) to be aware of the multicast address on which the replicator is transmitting. For the extension of the AMCA protocol, we have adopted an other approach based on the active networking technology. An active router maintains local information about the regulation tree and notifies the replicators to start the data replication. A replicator sends its flow toward its upstream active router which will forward every flow on the appropriate set of links.

The AMCA congestion avoidance protocol has been extended to implement the regulation tree approach using ns-2.1b8 (network simulator [2]). For practical considerations and ease of implementation, we chose to limit the number of subgroups maintained by an active router to 2. In this way, routers are not overloaded by the management of multiple subgroups. However, since every router performs locally its own partitioning procedure, the overall number of subgroups seen by the source can be much higher. For instance, if we consider a two-level multicast tree with N intermediate nodes (routers); if every router performs a 2-subgroup partition of its downstream receivers, we will get $(N+1)$ subgroups for the whole session.

4.1 Regulation Tree Construction within AMCA

An active router executes the partitioning algorithm proposed in section 3 and will split the set of its downstream links into two subgroups noted H and L which contain respectively the set of links with higher and lower capacity. Initially, and according to our partitioning algorithm, H contains all the downstream links, L is empty and the router is in the "initial phase". A router exits the initial phase when a partitioning is performed and a replicator is chosen. When a router receives a CR packet indicating a relative RTT variation greater than the b parameter, then it goes through the LS structure to determine if other links have experienced a relative RTT variation greater than the a threshold. If so all those links are moved from the H se to the L set. Once the two partitions H and L are built, the active router selects one link among those belonging to H as the replicator. Only direct receivers can be elected to act as a replicator by a router. This is why we have to be careful that the H set must contain at least one link leading directly to a receiver. To make a router aware of the links with direct receivers, the LS structure (introduced in section 2.1) contains a $flag$ field set to 1 if this link leads to one direct receiver and to 0 otherwise. Moreover in order to be able to evaluate the relative RTT variation experienced by every downstream link during the partitioning phase, the LS structure will contain the RTT, the RTT variation ($RTTvar$) and the period ($period$) reported by the last CR received on each link. For the replicator election, we use the following metric to evaluate the capacity of the candidate links (those leading directly to a receiver):

$$weight = \kappa\ RTTvar + (1-\kappa)\ \frac{1}{RTT \times (lr - lo + 1)} \tag{10}$$

where κ is a weighting parameter to be chosen in $]0,1[$. $RTTvar$ varies inversely with the available bandwidth on the considered link. The RTT gives an idea on the potential replicator distance to the active router. As for $(lr-lo+1)$, it can be seen as an estimation of the loss rate experienced by the receiver located downstream this link. Information on $RTTvar$, RTT, lr and lo are retrieved from the CRs (eventually from NACKs) received on every link.

Since we have limited the number of subgroups to only 2, we just need to maintain the identity of one replicator link in addition to the H and L contents for every subgroup. In order to make a replicator aware of its election, a data packet header is extended to contain an additional field ($IsRep$). Once a replicator is designated by a router, the first data packet received is forwarded on all the downstream links with the $IsRep$ field set to 1 only for the replicator link. Since an elected link leads directly to the replicator, only this latter will receive a data packet with this field set to 1. The reason behind sending this first data packet to all the links downstream is to make the replicator aware of its election. On the other hand this data packet, when forwarded on the L links, will have its $rate$ field set to half the source current rate. This is because the replicator will begin the replication from the next data packet but with a transmission rate set

to half the source current rate. This is important for setting the timeouts (especially for requesting repairs) at the receivers side.

Subsequent data packets received from the source are forwarded only on the H links. A replicator will retransmit every data packet it receives (with $IsRep$ set to 1) with an appropriate rate to its upstream router. This latter will retransmit the data packets received from downstream (from a replicator) only on the L links. For the receivers feedback, an active router forwards one aggregated CR based on the CRs received from the L links to the replicator. Similarly, the CRs received from the H receivers are aggregated into one CR to be sent upstream toward the source. The rate adjustment performed at the replicator side is the same as an AMCA sender (section 2.4). One difference consists in the fact that there is no slow-start and the replicator enters immediately in the congestion avoidance phase. Since its upstream router forwards feedbacks from the L receivers to it, the replicator would achieve a transmission rate which matches the slowest receiver among those located downstream the L links. The source will send data packet with a rate that matches the slowest receiver located downstream the H links.

We must mention that a replicator has to perform the cache of data packets to be retransmitted to its children. A replicator removes data packets from its cache as soon as it receives the corresponding acknowledgments (piggybacked in the CRs).

4.2 The Regulation Process at the Replicator's Side

A receiver, when it receives a data packets that indicates that it is the selected replicator, initializes a timer for each subsequent data packet it receives in order to schedule its retransmission to a time which corresponds to its regulation rate. For the data packet with sequence number i (see figure 7), a timer will be set depending on the desired reemission rate, to δ_i computed as follows:

$$\delta_i = \Delta + \delta_{i-1} - x_{i-1} \tag{11}$$

where x_{i-1} is the interval between the arrival of the $(i-1)$th and the ith data packets. x_{i-1} is inversely proportional to the source transmission rate. δ_{i-1} is the timeout value for the previous data packet. Δ is the reemission interval which is inversely proportional to the reemission rate. The replicator has to update the regulation timeout values every time the reemission rate changes.

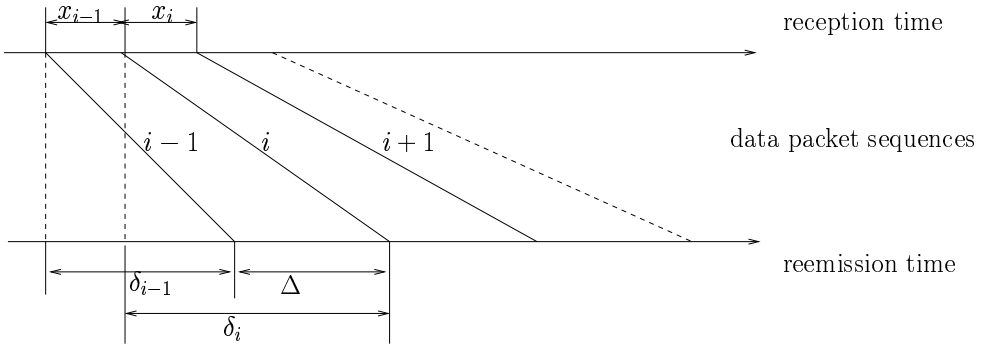


Figure 7: setting the reemission timers for the ith data packet

4.3 Partitioning Dynamics

We argue that our partitioning algorithm converges rapidly so that the initial partitioning is not disturbed by receivers changing their isolated rates. After the initial partitioning was performed, if any link experiences a RTT variation where $|\Delta\dot{\tau}| > b$ for a sufficiently long period[7], then a decision to move this receiver to the other subgroup could be taken.

[7] such a decision could be taken on the reception of K subsequent CRs with $|\Delta\dot{\tau}| > b$. K is a parameter is to be set appropriately to avoid oscillations while avoiding performances degradation

Moving a link from a subset to another does not require more than updating the L and H contents. The main concern is when the link leads to the replicator. If this link is the unique direct link in H, then a merge decision of the H and L subgroups can be taken. In this case, the router re-enters the initial phase and would eventually performs a new partitioning in the future depending on the network dynamics. Otherwise, the replicator is moved to L and a new replicator would be designated according to our capacity metric (10). Once done, the router will set the $isRep$ field to 1 when it forwards the subsequent data packet on the new replicator link. On the reception of this data packet, the old replicator knows that it is no longer the replicator and continues the replication of the standing packets. The new replicator, upon the reception of this data packet, would start its replication process as have been described.

5 Simulation Results

Using the ns-2 implementation of our 2-subgroup AMCA protocol, a set of simulations have been conducted. The original congestion parameters α, β, and N are respectively set to 1, 3 and 32. The partitioning parameters a, b and $\dot{\epsilon}$ are respectively set to 0.05, 0.2 and 0.25. The κ parameter is set to 0.5. For our simulations, we have adopted the topology of figure 8: one source S multicasts data packets to four receivers (with isolated rate $0.9Mbps$ for $R1$ and $R2$, $0.5Mbps$ for $R3$ and $R4$) through two active routers $A1$ and $A2$. The partitioning algorithm is enabled at the $A2$ router in order to increase the satisfaction of the four receivers. The simulation is run for 100 seconds of the real system.

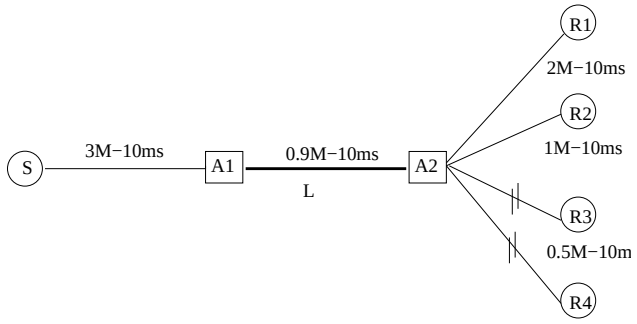


Figure 8: The used topology

Figure 9 shows the throughput achieved by the two subgroups $H = \{R1, R2\}$ and $L = \{R3, R4\}$ built by the active router $A2$. We can see that both the H and L receivers obtain their isolated rates of $0.9Mbps$ and $0.5Mbps$. Figure 10a shows the transmission rates of both the source and the chosen replicator (here $R1$). We can see that the source achieves rapidly a transmission rate of $900Kbps$, that the partitioning is performed in less than 2 seconds, and that the replicator achieves approximately a transmission rate of $500Kbps$ which corresponds to the isolated rate of the L receivers. Figure 10b shows the evolution of the data packet reception through their sequence numbers. At the beginning the two lines have the same slope. When the partitioning algorithm converges, one line keeps the same slope and the other achieves twice the original slope. Figure 11 shows the same experimentation conducted with the same topology but with an isolated rate of $1Mbps$ for the $R3$ and $R4$ and $2Mbps$ for the two others. Once again, the different receivers achieves a reception rate closest to their isolated rates.

6 Conclusion

In order to accommodate receivers heterogeneity for multicast communication in the Internet, we investigated a new multi-rate congestion mechanism. In a distributed fashion, every router performs a partitioning of its downstream links into subgroups of similar capacities and designates for each of them a receiver (called replicator) to perform data replication with the appropriate rate. The adopted partitioning algorithm is executed on-the-fly depending on the RTT variation feedbacks received by the routers thus avoiding a priori

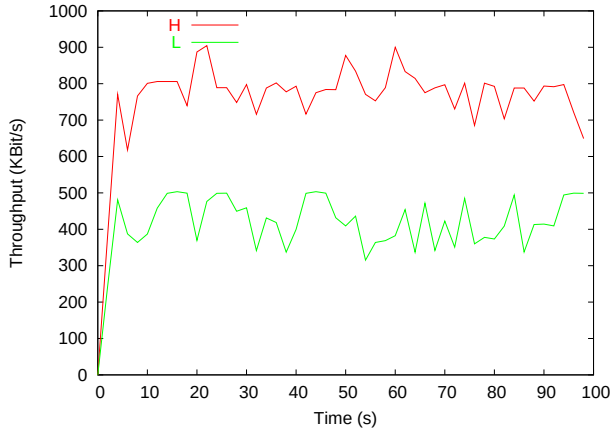
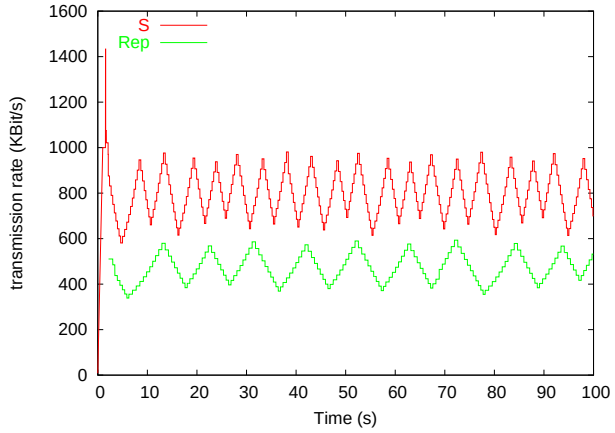
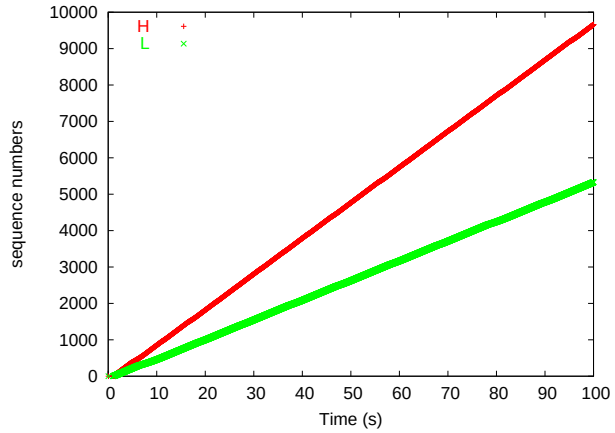


Figure 9: The 2 subgroups achieved hroughput.



(a)



(b)

Figure 10: (a) Transmission rate of the source and the replicator, (b) sequence numbers of data packets received by the H and the L receivers .

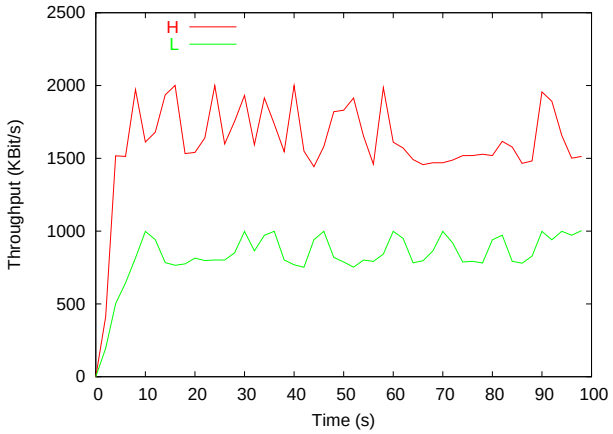


Figure 11: The 2 subgroups achieved hroughput.

estimation of the receivers' capacity. In addition to the construction of a regulation tree close to the physical multicast one, executing the partitioning algorithm at the routers instead of the source is more scalable.

To validate our approach, an AMCA (a single-rate congestion avoidance algorithm) extension is proposed where some practical issues have been considered. Preliminary simulation results show the rapid convergence of the partitioning process. As a future work, we plan to perform other simulations with more complex topologies mainly to evaluate the dynamic behavior of our approach in the case of receivers changing their capacities over the time.

7 Conclusion

References

[1] Dah Ming Chiu, Stephen Hurst, Miriam Kadansky, and Joseph Wesley. Tram: A tree-based reliable multicast protocol. Technical Report TR-98-66, SUN, July 1998.

[2] K. Fall, K. varadhan, and S. floyd. Ns notes and documentation ucb/lbnl/vint. software and documentation available at http://www.isi.edi/nsnam/ns, July 1999.

[3] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), 1997.

[4] V. Jacobson, S. McCanne, and M. Vetterl. Receiver-driven layered multicast. In *ACM SIGCOMM'96, Stanford, CA*, pages 117–130, August 1996.

[5] T. Jiang, M. Ammar, and E. Zegura. Inter-receiver fairness: A novel performance measure for multicast ABR sessions. In *Measurement and Modeling of Computer Systems*, pages 202–211, June 1998.

[6] T. Jiang, M. Ammar, and E. Zegura. On the use of destination set grouping to improve inter-receiver fairness for multicast abr sessions. In *IEEE INFOCOM'00*, March 2000.

[7] T. Jiang, E. Zegura, and M. Ammar. Inter-receiver fair multicast communication over the internet. In *the 9th International Workshopon Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 103–114, June 1999.

[8] S. Kasera and S. Bhattacharya. Scalabe fair reliable multicast using active services. *IEEE Network Magazine's Special Issue on Multicast*, 2000.

[9] L. Lehman, S. Garland, and D. Tennehouse. Active reliable multicast. In *Proc. of the IEEE INFOCOM, San Francisco, CA*, March 1998.

[10] X. Li, S. Paul, P. Pancha, and M.H. Ammar. Layered video multicast with retransmission (lvrm): Evaluation of hierarchical rate control. In *INFOCOM'97*, 1997.

[11] M. Maimour and C. Pham. Dyram : A reliable multicast protocol. Technical Report RR-4635, INRIA, 2002. Also available as a LIP/ENS Research Report under 2003-05.

[12] M. Maimour and C. Pham. Amca : an active-based multicast congestion avoidance algorithm. Technical Report RR-4689, INRIA, January 2003. Also available as a LIP/ENS Research Report under 2003-07.

[13] M. Maimour and C. Pham. A rtt-based partitioning algorithm for a multi-rate reliable multicast protocol. Technical Report 2003-16, LIP, March 2003.

[14] Christos Papadopoulos, Guru M. Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *Proc. of the IEEE INFOCOM*, March 1998.

[15] S. Paul and K. Sabnani. Reliable multicast transport protocol (RMTP). *IEEE JSAC, Spec. Issue on Network Support for Multipoint Communications*, 15(3), April 1997.

[16] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Conference on Computer Communications (IEEE Infocom'98), San Fransisco, USA*, pages 1–8, 1998.

[17] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, 1995.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399